In the Claims:

This listing of claims will replace all prior versions and listings of claims in the application.

Please amend claim 9 as follows:

1. (Original) A method of detecting viral code in subject files, comprising:

creating an artificial memory region spanning one or more components of the operating system;

emulating execution of computer executable code in a subject file; and

detecting when the emulated computer executable code attempts to access the artificial memory region.

2. (Original) The method of claim 1, wherein detecting when the emulated computer executable code attempts to access the artificial memory region comprises monitoring operating system calls by the emulated computer executable code.

3. (Original) The method of claim 1, further comprising:

determining an operating system call that the emulated computer executable code attempted to access; and

monitoring the operating system call to determine whether the computer executable code is viral.

4. (Original) The method of claim 1, further comprising:

determining an operating system call that the emulated computer executable code attempted to access; and emulating functionality of the operating system call while monitoring the operating system call to determine whether the computer executable code is viral.

5. (Original) The method of claim 1, further comprising monitoring accesses by the emulated computer executable code to the artificial memory region to detect looping.

6. (Original) The method of claim 1, wherein the artificial memory region spans an export table of one or more predetermined operating system components.

7. (Original) The method of claim 1, wherein creating an artificial memory region includes creating a custom version of an export table with predetermined values for the entry points.

8. (Original) The method of claim 1, further comprising monitoring access by the emulated computer executable code to dynamically linked functions.

9. (Amended) The method of claim 8, wherein the artificial memory region [created in step (a)] spans a jump table containing pointers to the dynamically linked functions.

10. (Original) A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for detecting viral code in subject files, the method steps comprising:

creating an artificial memory region spanning one or more components of the operating system;

emulating execution of computer executable code in a subject file; and detecting when the emulated computer executable code attempts to access the artificial memory region.

11. (Original) A computer system, comprising:

a processor; and

a program storage device readable by the computer system, tangibly embodying a program of instructions executable by the processor to perform method steps for detecting viral code in subject files, the method comprising:

creating an artificial memory region spanning one or more components of the operating system;

emulating execution of computer executable code in a subject file; and

detecting when the emulated computer executable code attempts to access the artificial memory region.

12. (Original) A computer data signal embodied in a transmission medium which embodies instructions executable by a computer for detecting in a subject file viral code that uses calls to an operating system, the signal comprising:

a first segment comprising CPU emulator code, wherein the CPU emulator code emulates execution of computer executable code in the subject file;

a second segment comprising memory manager code, wherein the memory manager code creates an artificial memory region spanning components of the operating system; and

a third segment comprising monitor code, wherein the monitor code detects when the emulated computer executable code attempts to access the artificial memory region.

13. (Original) The computer data signal of claim 12, further comprising:

a fourth segment comprising auxiliary code, wherein the auxiliary code determines an operating system call that the emulated computer executable code attempted to access; and

a fifth segment comprising analyzer code, wherein the analyzer code monitors the operating system call to determine whether the computer executable code is viral, while emulation continues.

14. (Original) An apparatus for detecting in a subject file viral code that uses calls to an operating system, comprising:

a CPU emulator;

a memory manager component that creates an artificial memory region spanning one or more components of the operating system; and

a monitor component, wherein the CPU emulator emulates execution of computer executable code in the subject file, and the monitor component detects when the emulated computer executable code attempts to access the artificial memory region.

15. (Original) The apparatus of claim 14, further comprising:

an auxiliary component; and

an analyzer component,

wherein the auxiliary component determines an operating system call that the emulated computer executable code attempted to access, and the analyzer component monitors the operating system call to determine whether the computer executable code is viral, while emulation continues.

16. (Original) The apparatus of claim 14, wherein the auxiliary component emulates functionalities of the operating system call.

17. (Original) The apparatus of claim 14, wherein the analyzer component monitors accesses by the emulated computer executable code to the artificial memory region to detect looping.

18. (Original) The apparatus of claim 14, wherein the artificial memory region created by the memory manager component spans an export table of one or more predetermined operating system components.

19. (Original) The apparatus of claim 14, wherein the memory manager component creates a custom version of an export table with predetermined values for the entry points.

20. (Original) The apparatus of claim 14, wherein the artificial memory region created by the memory manager component spans a jump table containing pointers to dynamically linked functions, and the monitor component monitors access by the emulated computer executable code to the dynamically linked functions.